Claims

1. A binary search tree comprising:

a multiplicity of address nodes each operable to store a data element, the nodes having pre-determined addresses and organised in a multiplicity of levels, the nodes including a root node and for each node at each level except the lowest level two child nodes in the immediately lower level, whereby the address of each child node is computable from the address of the respective node having that child node; and

a hardware engine for the insertion of elements in the nodes, said hardware engine being operable to make a search for the highest available node for the insertion of a new element and to search in a pattern in which all the nodes at each level beginning at the highest are searched before the search continues to the next lower level.

2. A binary search tree according to claim 1 wherein for each current node in said search the search comprises:

(a) determining whether a non-zero element is stored at the current node;

(b) determining, in the event that said non-zero element is stored, whether the current node is the last node at a current level of the tree;

(c) determining, if said current node is said last node, whether the current level is the lowest level of the tree;

(d) decrementing, if said current level is not the lowest level, the current level of the search and changing the current node to the first node of the next lower level of the tree; and

(e) setting, if said current node is not the last node at the current level, the current node to be the next node at the same level.

3. A binary search tree according to claim 2 wherein said engine, when the current node is available for the storage of a new element, causes the writing of a new element.

4. A binary search tree according to claim 2 wherein said engine, when said current node is available for the storage of a new element, is operative:

(i) to insert the new element if the current node is the root node, or when the current element is not the root node:

(ii) to determine whether the new element is greater or less than the element stored at the parent node of the current node and to increment or decrement the current node respectively; and

(iii) to insert the new element in accordance with an examination of the availability of the current node and a comparison of the magnitudes of the new element and the current element if any stored at the current node.

5. A method of establishing entries in a binary search tree, said binary search tree comprising a multiplicity of address nodes each operable to store a data element, the nodes having pre-determined addresses and organised in a multiplicity of levels, the nodes including a root node and for each node at each level except the lowest level two child nodes in the immediately lower level, whereby the address of each child node is computable from the address of the respective node having that child node:

said method comprising examining the nodes in a predetermined pattern to find a highest available node, said pattern requiring all the nodes at each level beginning at the highest to be examined before any node in the next lower level is examined.

6. A method according to claim 5 wherein said method comprises, for each current node that is examined:

(a) determining whether a non-zero element is stored at the current node:

(b) determining, in the event that said non-zero element is stored, whether the current node is the last node at a current level of the tree;

(c) determining, if said current node is said last node, whether the current level is the lowest level of the tree;

(d) decrementing, if said current level is not the lowest level, the current level of the search and changing the current node to the first node of the next lower level of the tree; and

(e) setting, if said current node is not the last node at the current level, the current node to be the next node at the same level.